# SISTEMA DE RESERVACIONES DE COWORKING

Desarrollar una API REST para un sistema de reservaciones de coworking. El proyecto consiste en crear un sistema de reservas en línea para un espacio de co-working. Los usuarios pueden registrarse, buscar espacios disponibles, reservar horas, y gestionar sus reservas. Además, el administrador del sistema puede agregar, actualizar, y eliminar espacios de co-working y revisar las reservas hechas por los usuarios.

# CARACTERISTICAS DEL PROYECTO

## Usuarios

* 1. Registro e inicio de sesión con JWT (JSON Web Tokens)
  2. Visualización de reservas realizadas

## Espacios de Co-Working

* 1. Realizar CRUD (crear, leer, actualizar, eliminar)
  2. Los espacios tendrán detalles como nombre, ubicación, precio por hora, capacidad, y disponibilidad.

## Sistema de Reserva

* 1. Los usuarios pueden buscar espacios disponibles y seleccionarlos
  2. Gestión de reservas

## Administradores

* 1. Inicio de sesión para administradores
  2. Visualizar y gestionar todos los espacios de coworking

# ENDPOINTS

## Auth

* + - 1. [POST] /auth/user/login: Login de usuarios
      2. [POST ]/auth/admin/login: Login de administradores
      3. [POST] /auth/user/register: Registro de usuarios

## Coworking (Solo administradores pueden acceder)

* + - 1. [GET] /coworking: Obtener todos los coworking
      2. [GET] /coworking/{id}: Obtener coworking por id
      3. [POST] /coworking: Crear coworking
      4. [PUT] /coworking/{id}: Actualizar coworking por id
      5. [DELETE] /coworking/{id}: Eliminar coworking por id

## Reservation

* + - 1. [GET] /reservation: Listar todas las reservaciones
      2. [GET] /reservation/{id}: Obtener reservación por id
      3. [POST] /reservation: Crear reservación
      4. [PUT] /reservation/{id}: Actualizar reservacion por id
      5. [DELETE] /reservation/{id}: Eliminar reservacion por id

# IMPORTANTE:

* + - * + Solo los administradores tienen acceso a las rutas de coworking por lo que se debe validar el role.
        + Las rutas de **coworking** y **reservation** deben estar protegidas con un JWT, por lo que cada vez que se quiera acceder a una ruta se debe enviar y validar el JWT
        + En el siguiente proyecto muestro como utilizar el JWT: <https://github.com/itsronalds/CoworkReserveAPI>
        + Usuarios interactuarán con los endpoints de reservation, y administradores con los endpoints de coworking

# MODELO DE BASE DE DATOS SQL

## Usuario

* + - * + Id Int (Primary Key)
        + Name Varchar (127) NN
        + Email VARCHAR (127) NN
        + Password VARCHAR (127) **Hashed** NN
        + CreatedAt TIMESTAMP (Fecha de registro)

## Coworking

* + - * + Id Int (Primary Key)
        + Name Varchar (127) NN
        + Location VARCHAR (127) NN
        + PriceByHour DECIMAL (10, 2) NN
        + Capacity INT NN
        + IsAvailable INT NN

## Reservation

* + - * + Id INT (Primary Key)
        + UserID INT NN
        + CoworkingID INT NN
        + Date Date NN
        + StartTime TIME NN
        + EndTime NN

## Admin

* + - * + Id INT (Primary Key)
        + Name VARCHAR (127) NN
        + Email VARCHAR (127) NN
        + Password VARCHAR (127) **Hashed** NN

# TECNOLOGIAS

* + - * + **FastAPI**: Para la creación de los endpoints de la API y la gestión de la autenticación
        + **MySQL**: Para la base de datos relacional, manejando la persistencia de usuarios, espacios y reservas
        + **SQLAlchemy**: Como ORM para interactuar con la base de datos
        + **Pydantic**: Para la validación de los datos de entrada/salida

# CREDENCIALES

Solo hay un administrador registrado y solo con ese se trabajará, sus credenciales de login son:

**Email**: [admin@gmail.com](mailto:admin@gmail.com)

**Password**: 123456789

# FECHA DE ENTREGA

28-09-2024 12PM